CLAIMS

What is claimed is:

1. A method for designing a software architecture for utilizing software components in building N-tier software applications, the method comprising:

a. specifying a set of software component rules for creating software components;

b. specifying a set of tier rules for creating an extensible set of tiers, the tier rules further comprising:

i. a set of association rules by which at least one software component created using the software component rules may be associated with or disassociated from at least one tier created with the set of tier rules;

ii. a set of tier framework rules to provide an architected context for software components associated with a tier; and

iii. a set of package rules to provide for logical grouping of interfaces within a framework defined by the tier framework rules to provide a set of specific behaviors for the tier; and

c. specifying a set of assembly rules, the assembly rules comprising association rules by which each tier may be associated with at least one other tier and linkage rules by which each tier may be linked to at least one other tier.

2. The method of claim 1 wherein specifying a set of software component rules for creating software components further comprises:

a. specifying rules for specifying interfaces for each software component; and

b. specifying rules for specifying behavior exhibited by each software component.

3. The method of claim 2 wherein specifying rules for specifying behavior further comprises:

a. specifying rules on how each software component encapsulates details of how functionality is implemented for that software component; and

b. specifying rules on creating a well-defined interface reusable at a binary level for each software component;

c. whereby each software component may be made available for use by any other software component that can use the well-defined interface of the first software component.

4. The method of claim1 further comprising specifying library rules for selectively placing software components into and retrieving software components from an inventory of software components.

5. The method of claim 1 wherein software component rules comprise rules for supporting off-the-shelf components within software components or tiers, including rules to allow addition of off-the-shelf software components into the inventory.

6. The method of claim 1 further comprising specifying at least one software component modification rule whereby software components may be extended, the at least one software component modification rule comprising addition, modification, and deletion rules.

7. The method of claim 1 wherein the software component rules further comprise rules for designating software component function points.

8. The method of claim 7 wherein the rules for designating software component function points further comprise rules to allow implementing software component interfaces required by a particular tier to which the software component belongs.

9. The method of claim 1 wherein specifying a set of tier rules for creating an extensible set of tiers further comprises:

a. specifying rules on allowing modification of software component attributes for software components associated with the tier;

b. specifying rules to allow specifying what dependencies a framework has to other frameworks;

c. specifying rules on how properties and interfaces are grouped;

d. specifying rules on what interfaces are used; and

e. specifying rules that specify where software component behavior belongs.

10. The method of claim 1 wherein the framework rules further comprise rules on specifying at least one package for a framework, the package further comprising a set of interfaces to provide a specific behavior.

11. The method of claim 1 further comprising:

a. specifying a basic design structure comprising base components for software components in the tier; and

b. specifying a set of standard interfaces for the software components categorized as belonging to the tier.

12. The method of claim 1 wherein specifying a set of assembly rules further comprises:

a. specifying rules to allow assembling and compiling at least one tier to provide a stand-alone, executable program; and

b. specifying rules on allowing combining software components and invoking an assembled application at run-time to form new unique applications on-the-fly.

13. The method of claim 1, wherein the software component rules specify rules allowing each software component to execute asynchronously within its own thread and time frame and to inform dependent components of its status or provide dependent components with information when predetermined events occur.

14. The method of claim 1 further comprising specifying rules to allow defining one or more techniques to allow a software component to traverse a model, the model comprising one or more software components.

15. The method of claim 14 wherein the traversal of a model uses a predetermined interface.

16. The method of claim 1 further comprising specifying rules on implementing a template iterator class to facilitate accessing associations.

17. The method of claim 16 wherein the template iterator class can be based at any software component's associations and can be used to iterate through all software components in the association or only through a specific software component type.

18. A method for generating software components for use in an N-tier software application, the software components having a predetermined structure, the method comprising:

a. providing a software component architecture comprising a plurality of tiers, wherein each tier may be associated with at least one of the software components, each tier further comprising a predetermined set of interfaces for that tier, the interfaces defining a set of functionality capable within that tier; and

b. for a selected one of the plurality of tiers, providing at least one of the software components to satisfy the functionality of the tier wherein the at least one of the software components provides a predetermined set of interfaces specified for the selected one of the plurality of tiers.

19. The method of claim 18 wherein the software component is reusable by any system employing software components designed in accordance with the method of claim 18.

20. The method of claim 18, further comprising:

a. specifying a set of tier framework rules to provide an architected context for software components within a tier; and

b. specifying a set of package rules to provide for logical grouping of interfaces within a framework defined by the tier framework rules to provide a set of specific behaviors for the tier.

21. The method of claim 20, wherein the tier framework rules further comprise:

a. specifying rules on specifying dependencies a framework has to other frameworks;

b. specifying how properties and interfaces are grouped;

c. specifying what interfaces are used; and

d. specifying where software component behavior belongs.

22. The method of claim 21, further comprising specifying rules on defining packages, the packages comprising grouping of interfaces within a framework into subsets of interfaces to specify how a specific behavior, such as messaging or connecting, is to be provided.

23. A method of system design for an N-tier architecture, the architecture comprising software components and tiers, the method comprising:

a. determining a set of application requirements;

b. determining a list of required models and software components to satisfy the application requirements;

c. logically grouping the software components into extensible tiers;

d. determining if each software component in each tier is available in an inventory of components;

e. using each software component found in the inventory if that software component is a required software component;

f. restructuring software components in the inventory to ensure conformance while retaining the original intent of the requirement, if possible;

g. adding additional software components if no existing software component in the inventory satisfies or can be restructured to satisfy a requirement;

h. associating at least one software component with each required tier; and

i. creating an application by defining and implementing linkages between the required tiers.

24. The method of claim 23 further comprising:

a. testing each new software component;

b. testing each restructured software component;

c. assessing each new software component for suitability to become part of the software inventory;

d. assessing each restructured software component for suitability to become part of the software inventory; and

e. adding the new or restructured components to the inventory if the new or restructured software component has potential for reuse are added to the software inventory.

25. The method of claim 23 wherein the new or restructured software components are either tailored into the current architecture or the architecture is expanded by adding one or more tiers to accommodate the new or restructured software component.

26. The method of claim 23 where software components that are so specific they can only be used in a current application are not added to the inventory.

27. The method of claim 24 wherein testing comprises testing and validation.

28. The method of claim 23 wherein adding additional software components if no existing software component in the inventory satisfies or can be restructured to satisfy a requirement further comprises:

a. procuring an off-the-shelf software component from a third party; and

b. providing the off-the-shelf software component with a predetermined interface to interface between the off-the-shelf software component and at least one tier.

29. A system for designing a software architecture for use in generating software components for building software applications, the system comprising:

a. at least one processing unit;

b. at least one memory store operatively connected to the processing unit;

c. N-tier design software executable within the at least one processing unit;

d. software architecture specifications resident in the memory store for use by the N-tier design software, the software architecture specifications comprising specifications for a set of software component rules for creating software components, specifications of a set of tier rules for creating tiers, and specifications of a set of assembly rules;

e. an input device, operatively in communication with the processing unit, for permitting input of the software architecture specifications;

f. an output device, operatively in communication with the processing unit; and

g. a communications pathway operatively connected to the processing unit.

30. The system of claim 29 wherein the communications pathway is a network.

31. The system of claim 30 wherein the network comprises asynchronous communications, synchronous communications, local communications, local area networks, wide area networks, and local bus networks.

32. A system for designing a software architecture for use in generating software components for building software applications, the system comprising:

a. means for specifying a set of software component rules for creating software components;

b. means for specifying a set of tier rules for creating tiers, the tier rules further comprising:

i. a set of association rules by which each tier created with the set of tier rules may be associated with at least one software component created using the software component rules;

ii. a set of tier framework rules to provide an architected context for software components within a tier; and

iii. a set of package rules to provide for logical grouping of interfaces within a framework defined by the tier framework rules to provide a set of specific behaviors for the tier; and

c. means for specifying a set of assembly rules further comprising association rules by which each tier may be associated with at least one software component and linkage rules by which each tier may be linked to at least one other tier.

33. A method for defining and implementing an N-tier software architecture for a system comprising at least one processing unit, at least one memory store operatively connected to the processing unit, N-tier designing software executable within the at least one processing unit, an input device operatively in communication with the processing unit for permitting input of the software architecture specifications, an output device operatively in communication with the processing unit, and a communications pathway operatively connected to the processing unit, the method comprising:

a. loading the N-tier designing software into the memory store;

b. executing the N-tier designing software;

c. inputting a set of software component rules for creating software components into the memory store;

d. inputting a set of tier rules for creating tiers into the memory store, the tier rules further comprising:

i. a set of association rules by which each tier created with the set of tier rules may be associated with at least one software component created using the software component rules;

ii. a set of tier framework rules to provide an architected context for software components within a tier; and

iii. a set of package rules to provide for logical grouping of interfaces within a framework defined by the tier framework rules to provide a set of specific behaviors for the tier;

e. inputting a set of assembly rules into the memory store, the assembly rules further comprising association rules by which each tier may be associated with at least one software component and linkage rules by which each tier may be linked to at least one other tier; and

f. processing the software component rules, tier rules, and assembly rules using the N-tier designing software to create an N-tier software architecture.

34. An N-tier software architecture stored in a storage media, the storage media comprising:

a. a first plurality of binary values for creating software components using software component rules;

b. a second plurality of binary values for creating tiers using tier rules; and

c. a third plurality of binary values for assembling software applications from tiers and software components.